Requested Patent: JP5265838A

Title:

OBJECT-ORIENTATION DATA BASE SYSTEM AND VERSION MANAGING METHOD

Abstracted Patent: JP5265838

Publication Date: 1993-10-15

Inventor(s): NAMIOKA MIYOKO; others: 05

Applicant(s): HITACHI LTD

Application Number: JP19920090246 19920316

Priority Number(s):

IPC Classification: G06F12/00 ; G06F9/44 ; G06F12/00

Equivalents:

ABSTRACT:

PURPOSE:To provide the object-orientation data base system and version managing method reducing memory waste and the reduction of efficiency for executing the procedure of user definition.

CONSTITUTION:An attribute to store a pointer to a meta version is defined as an instance parameter, a version-made object class 51 to hold a method for writing and reading values to the attribute is held as a system library 50, and the definition of the object class made to be a version 51 is succeeded by the class of a compile request user. Respective attributes for storing the pointer to a set object having the plural versions of the object as elements, for storing the pointer to the representative version of elements in the set object and for storing the pointer to generic instance are defined as instance parameters, and a meta version class 52 to hold a method for writing and reading values to the respective attributes and a version manager class 53 to hold the version making means of the object as a method are held as the library 50.

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（A）**　(11)特許出願公開番号

特開平5－265838

(43)公開日　平成５年(1993)10月15日

(51)Int.Cl.[5]	識別記号	庁内整理番号	Ｆ１	技術表示箇所
Ｇ０６Ｆ　12/00	５４７　Ａ	7232－5B		
9/44	３３０　Ｚ	9193－5B		
12/00	５１７	7232－5B		

審査請求　未請求　請求項の数６（全 22 頁）

(21)出願番号　特願平4－90246

(22)出願日　平成４年(1992)３月16日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　浪岡　美予子
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(72)発明者　山本　洋一
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(72)発明者　浅見　真人
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(74)代理人　弁理士　笹岡　茂　（外１名）

最終頁に続く

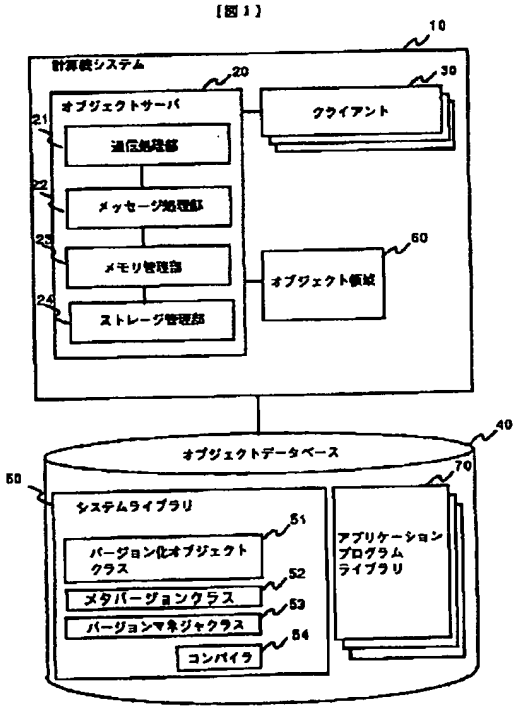
(54)【発明の名称】　オブジェクト指向データベースシステム及びそのバージョン管理方法

(57)【要約】

【目的】　メモリ浪費及びユーザ定義の手続き実行効率の低下の少ないオブジェクト指向データベースシステム及びバージョン管理方法

【構成】　メタバージョンへのポインタを格納する属性をインスタンス変数とし、該属性への値書込み用と値読込用メソッドを保持するバージョン化オブジェクトクラスをライブラリとして保持し、コンパイル要求ユーザのクラスに対しバージョン化オブジェクトクラスの定義を継承させ、オブジェクトの複数のバージョンを要素とする集合オブジェクトへのポインタ格納用、該集合オブジェクトの要素を代表する代表バージョンへのポインタ格納用、ジェネリックインスタンスへのポインタ格納用の各属性をインスタンス変数とし、各属性への値書込用と値読込用メソッドを保持するメタバージョンクラスと、オブジェクトのバージョン化手段をメソッドとして保持するバージョンマネジャクラスとをライブラリとして保持する。

【図１】



1

【特許請求の範囲】

【請求項1】　オブジェクトに対する複数のバージョンを保持するオブジェクト指向データベースシステムであって、

オブジェクト指向データベース中のオブジェクトについて、当該オブジェクトのバージョン管理情報を保持するメタバージョンへのポインタを格納する属性を付加する手段と、当該属性に値を書き込む手段と、当該属性の値を読み込む手段と、

前記メタバージョンについては、前記オブジェクトの複数のバージョンを代表する唯1つの代表バージョンへのポインタを格納する属性を付加する手段と、当該属性に値を書き込む手段と、当該属性の値を読み込む手段と、

与えられた1組のオブジェクトとメタバージョンより、付与された該オブジェクトを複写して該オブジェクトの最初のバージョンとなるオブジェクトを生成し、複写元の前記オブジェクトに該オブジェクトがジェネリックインスタンスであることを示すタグを付加し、該オブジェクトと該オブジェクトの前記最初のバージョンについて、それぞれのメタバージョンへのポインタを格納する属性に前記付与されたメタバージョンへのポインタを示す値を書き込むオブジェクトのバージョン化手段と、

オブジェクトに付加されたメタバージョンへのポインタを格納する属性の値、該メタバージョンが保持する該オブジェクトの代表バージョンへのポインタを格納する属性の値ならびに該オブジェクトがジェネリックインスタンスであることを示すタグの有無に基いて、該オブジェクトへ送信されたメッセージを処理する手段、を有することを特徴とするオブジェクト指向データベースシステム。

【請求項2】　請求項1記載のオブジェクト指向データベースシステムにおいて、前記メタバージョンへのポインタを格納する属性をインスタンス変数とし、当該属性に値を書き込む手段ならびに当該属性の値を読み込む手段をメソッドとして保持するバージョン化オブジェクトクラスをライブラリとして保持し、コンパイルを要求するユーザのクラスに対して当該バージョン化オブジェクトクラスの定義を継承させることを特徴とするオブジェクト指向データベースシステム。

【請求項3】　請求項1記載のオブジェクト指向データベースシステムにおいて、前記オブジェクトの複数のバージョンを要素とする集合オブジェクトへのポインタを格納する属性と、該集合オブジェクトの要素である複数のバージョンを代表する唯1つの代表バージョンへのポインタを格納する属性と、前記ジェネリックインスタンスへのポインタを格納する属性をインスタンス変数とし、それぞれの属性に値を書き込む手段ならびに値を読み込む手段をメソッドとして保持するメタバージョンクラスをライブラリとして保持することを特徴とする請求項1記載のオブジェクト指向データベースシステム。

2

【請求項4】　請求項1記載のオブジェクト指向データベースシステムにおいて、前記オブジェクトのバージョン化手段をメソッドとして保持するバージョンマネジャクラスをライブラリとして保持することを特徴とする請求項1記載のオブジェクト指向データベースシステム。

【請求項5】　請求項1記載のオブジェクト指向データベースシステムにおいて、

前記メタバージョンへのポインタを格納する属性をインスタンス変数とし、当該属性に値を書き込む手段ならびに当該属性の値を読み込む手段をメソッドとして保持するバージョン化オブジェクトクラスをライブラリとして保持し、コンパイルを要求するユーザのクラスに対して当該バージョン化オブジェクトクラスの定義を継承させ、

前記オブジェクトの複数のバージョンを要素とする集合オブジェクトへのポインタを格納する属性と、該集合オブジェクトの要素である複数のバージョンを代表する唯1つの代表バージョンへのポインタを格納する属性と、前記ジェネリックインスタンスへのポインタを格納する属性をインスタンス変数とし、それぞれの属性に値を書き込む手段ならびに値を読み込む手段をメソッドとして保持するメタバージョンクラスをライブラリとして保持し、

前記オブジェクトのバージョン化手段をメソッドとして保持するバージョンマネジャクラスをライブラリとして保持することを特徴とするオブジェクト指向データベースシステム。

【請求項6】　オブジェクト指向データベースシステムにおけるバージョン管理方法であって、

当該オブジェクト指向データベース中のオブジェクトに、該オブジェクトのバージョン管理情報を保持するメタバージョンへのポインタを格納する属性を持たせ、該メタバージョンには、前記オブジェクトのバージョンを要素とする集合オブジェクトへのポインタを格納する属性ならびに該集合オブジェクトの要素を代表する代表バージョンへのポインタを格納する属性を持たせておき、

オブジェクトの生成要求があった場合には、該要求により生成したオブジェクトの前記メタバージョンへのポインタを格納する属性に、該メタバージョンが存在しない旨を設定しておき、

該オブジェクトのバージョン化要求があった場合には、該オブジェクトに対するメタバージョンを生成し、該オブジェクトを複写して該オブジェクトの最初のバージョンを生成し、該オブジェクトにはジェネリックインスタンスであることを示すタグを付加し、該オブジェクトならびに該バージョンのメタバージョンへのポインタを格納する属性には、該オブジェクトに対して生成しておいた前記メタバージョンへのポインタを示す値を設定し、

当該メタバージョンについては、該バージョンを要素とする集合オブジェクトを生成して、当該集合オブジェク

3

トへのポインタを示す値と該バージョンへのポインタを示す値を、それぞれ、バージョンを要素とする集合オブジェクトへのポインタを格納する属性、集合オブジェクトの要素を代表する代表バージョンへのポインタを格納する属性に設定しておき、

該オブジェクトに新たにバージョンを追加する場合は、該オブジェクトのメタバージョンに設定されている集合オブジェクトに、当該バージョンを要素として追加し、

該オブジェクトの代表バージョンを変更する場合は、該オブジェクトのメタバージョンに設定されている代表バージョンへのポインタを、新たな代表バージョンへのポインタに書き換え、

メッセージを処理する場合には、そのメッセージのレシーバに指定されているオブジェクトについて、該オブジェクトがジェネリックインスタンスであることを表すタグがあるか否かを判定し、該オブジェクトがジェネリックインスタンスでない場合は、該オブジェクトをレシーバとしてメッセージを処理し、該オブジェクトがジェネリックインスタンスである場合は、該オブジェクトのメタバージョンを得、該メタバージョンから代表バージョンを得、当該代表バージョンをレシーバとしてメッセージを処理することを特徴とするバージョン管理方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】オブジェクト指向データベースシステムと、そのバージョン管理方法に関する。

【０００２】

【従来技術】最近、ＥＣＡＤ（ｅｌｅｃｔｒｉｃａｌ ｃｏｍｐｕｔｅｒ ａｉｄｅｄ ｄｅｓｉｇｎ）、ＭＣＡＤ（ｍｅｃｈａｎｉｃａｌ ｃｏｍｐｕｔｅｒ ａｉｄｅｄ ｄｅｓｉｇｎ）、ＳＣＡＤ（ｓｏｆｔｗａｒｅ ｃｏｍｐｕｔｅｒ ａｉｄｅｄ ｄｅｓｉｇｎ）などのアプリケーションをサポートすることが、データベースシステムの主要な課題となっている。

【０００３】前記アプリケーションが扱うデータは、構造が複雑である、種類が多い、また異なった種類のデータは異なった手続きを必要とする等の特徴を持っている。

【０００４】このようなデータを扱うためには、現在一般的になりつつあるリレーショナルデータモデルなどでは表現能力が不足している。これに対して、近年データの意味表現能力の拡大を目的に種々のデータモデルや、それに基づくシステムの研究がなされている。特に、オブジェクト指向の考えにデータ意味論的な概念を融合させたオブジェクト指向データモデルの有用性が注目されている。

【０００５】オブジェクト指向データモデルでは、実世界のデータを全てオブジェクトとして扱い、それらに対して統一的な操作インタフェースを与える。また、オブジェクト間の汎化・専化、集約などの関係の表現を可能

4

にし、実世界のデータの持つ意味をデータベース上に表現する能力が従来のデータベースに比べ優れている。このようなオブジェクト指向データモデルに基づくデータベースシステムを、オブジェクト指向データベースシステムと呼んでいる。

【０００６】上記にあげた応用分野では、設計に代表される試行錯誤的な作業がなんらかのバージョン機能を必要とする。そのため、バージョン管理機能は、オブジェクト指向データベースがサポートすべき重要な機能の１つとなっている。バージョン管理でサポートしなければならない基本的な機能としては、オブジェクトのバージョン化を可能とすること、バージョンに対する暗黙的参照と明示的参照を可能とすること、バージョン間の関連を管理することを挙げることができる。以下、それぞれの機能について簡単に説明する。

【０００７】通常、オブジェクトについては、それを操作するためのインタフェースが使用者に対して公開されるが、そのインタフェースを実現するためのインプリメンテーション部分は使用者から隠されている。このインプリメンテーション部分は、オブジェクトの状態を格納するための変数（インスタンス変数）と手続き（メソッド）を含んでいる。オブジェクトの状態とは、オブジェクトが持っている変数の値によって決まる。オブジェクトのバージョン管理機能を持たないシステムでは、オブジェクトは唯１つの状態のみを持つ。オブジェクトのバージョン化を可能にするとは、オブジェクトが複数の状態を持つことを可能にするということである。オブジェクトＡについて、オブジェクトＡの異なった状態を表すオブジェクトをオブジェクトＡのバージョンと呼ぶ。バージョンという言葉に対し、それらが状態を表しているオブジェクトをジェネリックインスタンスと呼ぶことが多い。

【０００８】オブジェクト指向データベースでは、オブジェクトに対してそれを一意に識別することができるオブジェクト識別子を与えるのが普通である。オブジェクトを参照する場合、このオブジェクト識別子を用いることができる。バージョンの明示的参照を可能にするとは、あるジェネリックインスタンスの任意のバージョンを直接参照可能にすることである。また、バージョンの暗黙的参照を可能にするとは、ジェネリックインスタンスへの参照に対して、実際にその具体的な状態が必要になったときに、その参照を特定のバージョンへ動的に結合することである。どちらの参照を用いた場合でも、同じジェネリックインスタンスに対応するバージョンは、各々の状態は異なっていても、ジェネリックインスタンスと同じインタフェースを持たなければならない。

【０００９】１つのジェネリックインスタンスの複数のバージョンの間には、生成に関する時系列的な関係や親子関係などが存在するのが普通である。バージョン間の関係を管理するとは、このような関係を維持し、操作す

5

ることを可能にすることである。上記のようなバージョン管理機能をサポートする方法として、バージョン固有の属性や手続きを定義したクラスをオブジェクト指向データベースシステムが用意し、その定義をユーザが定義したクラスに継承させるという方法が、以下の論文（1）（2）（3）の中で論じられている。

【0010】（1）R. Agrawal et. al.,"Object Versioning in Ode", Proc. IEEE Data Engineering, 1991, 446－455.

文献（1）のシステムでは、該システムが提供するプログラミング言語で記述されたユーザ定義のクラスは、該システムのコンパイラによって、汎用クラスVersionの定義を継承するようなコードに変換される。このVersionというクラスは、バージョンの時系列リスト、バージョンの導出木、バージョンから成る集合オブジェクトへのポインタを保持する変数を含み、バージョンの追加や削除、バージョンから成る集合オブジェクトへのポインタを返す手続き等を含んでいる。

【0011】以下の文献（2）のシステムや文献（3）のシステムでは、システムが付加する属性や手続きが、ジェネリックインスタンスとバージョンで異なるという点で文献（1）のシステムとは少し違っている。しかし、オブジェクトがユーザ定義の属性や手続きの他に、システムが定めたバージョン管理のための属性や手続きを持つということは同じである。

（2）H. T. Chou and W. Kim,"VERSION AND CHANGE NOTIFICATION IN AN OBJECT-ORIENTED DATABASE SYSTEM", Proc. 25th ACM/IEEE Data Automation Conf., 1988, 275－281.

（3）D. Beech and B. Mahbod,"GeneralizedVersion Control in an Object-Oriented Database", Proc. IEEE 4th Int'l Conf. Data Engineering, 1988, 14－22.

次の論文の中で提案されている方法は、上記とは少し異なる。

【0012】（4）Lichao Tan, et. ai.,"Attribute Grammar Based Structure Evolution in Object Management System OOAG", 1990, 情報処理学会研究報告Vol. 90, No. 68.

ここでは、オブジェクトに対して、そのオブジェクトのバージョンに関する情報を一手に維持・管理するオブジェクト（これを、メタオブジェクトと呼んでいる）について論じられている。オブジェクトが生成されるとシステムが自動的に対応するメタオブジェクトを生成する。オブジェクトに送信されたメッセージは、全てシステムがそのオブジェクトのメタオブジェクトに転送する。そして、バージョンに関するメッセージの場合は、このメタオブジェクトが処理する。

6

【0013】

【発明が解決しようとする課題】文献（1）～（3）のように、ユーザ定義の属性や手続きの他に、バージョン管理に必要な属性や手続きをオブジェクトに付加するという方法は、次のような問題がある。

（1）付加される属性の数だけ、オブジェクトのサイズが大きくなるため、バージョン管理を必要としないオブジェクトについては、バージョン管理のための余分な属性を持つことになりメモリを浪費する。

（2）バージョン管理のために付加される手続きの数だけ、手続き探索に余分な時間がかかるため、ユーザ定義の手続きの実行性能を低下させることになる。

【0014】このような問題点について文献（2）では、予めクラス単位でバージョン管理を必要とするオブジェクトか否かを定義しておくことにより、問題（1）（2）によるオーバヘッドを局所化することが提案されているが、この方法では、クラス定義時にそのクラスのインスタンスについて、バージョン管理の必要の有無を決定しなければならないため、使い勝手の面で問題がある。また、バージョン管理の対象となるクラスの全てのインスタンスがバージョン管理を必要とするわけではないから、そのようなインスタンスについてのオーバッヘッドは残る。文献（3）では、クラス単位ではなく個々のオブジェクトについて、バージョンを持たない状態からバージョンを持った状態へ変換する操作を設けることにより、問題（1）を回避している。文献（3）では、永続性を持つクラスのオブジェクトのみが、バージョン管理の対象となるため、文献（2）と同様、問題はそのようなクラスにのみ局所化される。

【0015】文献（4）による方法は、オブジェクトに付加する属性や手続きが文献（1）～（3）の方法に比べて少なくて済む。しかし、本来オブジェクト自身で処理できるユーザ定義の手続きまでが、メタオブジェクトに転送されるため、手続きの実行性能を低下させることになる。

【0016】従来技術にあげたシステムにおけるバージョン管理方法は、先に述べた基本的なバージョン管理機能、すなわち、オブジェクトのバージョン化、バージョンへの明示的参照と暗黙的参照、バージョン間の関連の管理、などを実現することができる。しかし、オブジェクトの改訂の際の処理方法やバージョン間の関連の付け方など、実際のバージョン管理は、適用分野ごとに多種の方法が考えられる。適用分野にマッチしたバージョン管理機能を実現するには、バージョン管理のためのデータ構造を利用者がクラスとして定義し、バージョン管理

操作をメソッドとして記述できることが望ましい。従来例にあげた方法は、このような要求との親和性が低い。

【0017】本発明の目的は、オブジェクト指向データベースシステムにおいて、メモリの浪費ならびにユーザ定義の手続きの実行効率の低下の少ないデータベースシステムおよびバージョン管理方法を提供することである。本発明の他の目的は、オブジェクト指向データベースシステムにおいて、適用分野にマッチしたバージョン管理機能の実現を容易にするデータベースシステムおよびバージョン管理方法を提供することである。

【0018】

【課題を解決するための手段】上記課題を解決するため、本発明のオブジェクト指向データベースシステムでは、オブジェクト指向データベース中のオブジェクトに対して、当該オブジェクトのバージョン管理情報を保持するメタバージョンへのポインタを格納する属性を付加する手段と、当該属性に値を書き込む手段と、当該属性の値を読み込む手段と、上記メタバージョンについては、上記オブジェクトの複数のバージョンを代表する唯1つの代表バージョンへのポインタを格納する属性を付加する手段と、当該属性に値を書き込む手段と、当該属性の値を読み込む手段と、与えられた1組のオブジェクトとメタバージョンより、該オブジェクトを複写して該オブジェクトの最初のバージョンとなるオブジェクトを生成し、複写元のオブジェクトに該オブジェクトがジェネリックインスタンスであることを示すタグを付加し、該オブジェクトと該オブジェクトの上記最初のバージョンについて、それぞれのメタバージョンへのポインタを格納する属性に上記付与されたメタバージョンへのポインタを示す値を書き込むオブジェクトのバージョン化手段と、オブジェクトに付加された、メタバージョンへのポインタを格納する属性の値、該メタバージョンが保持する該オブジェクトの代表バージョンへのポインタを格納する属性の値、ならびに該オブジェクトがジェネリックインスタンスであるか否かを示すタグの有無に基いて、該オブジェクトへ送信されたメッセージを処理する手段とを有する。

【0019】特に本発明の望ましい態様では、上記メタバージョンへのポインタを格納する属性をインスタンス変数とし、当該属性に値を書き込む手段ならびに当該属性の値を読み込む手段をメソッドとして保持するバージョン化オブジェクトクラスをライブラリとして保持し、コンパイルを要求するユーザのクラスに対して当該バージョン化オブジェクトクラスの定義を継承させ、上記オブジェクトの複数のバージョンを要素とする集合オブジェクトへのポインタを格納する属性と、該集合オブジェクトの要素である複数のバージョンを代表する唯1つの代表バージョンへのポインタを格納する属性と、上記ジェネリックインスタンスへのポインタを格納する属性をインスタンス変数とし、それぞれの属性に値を書き込む手段ならびに値を読み込む手段をメソッドとして保持するメタバージョンクラス、与えられた1組のオブジェクトとメタバージョンに基いて、付与されたオブジェクトを複写して該オブジェクトの最初のバージョンとなるオブジェクトを生成し、該オブジェクトに該オブジェクトがジェネリックインスタンスであることを示すタグを付加し、該オブジェクトと該オブジェクトの上記最初のバージョンについて、それぞれのメタバージョンへのポインタを格納する属性に上記付与されたメタバージョンへのポインタを示す値を書き込むオブジェクトのバージョン化手段をメソッドとして保持するバージョンマネジャクラスをライブラリとして保持する。

【0020】また、本発明のバージョン管理方法は、オブジェクト指向データベース中のオブジェクトに、該オブジェクトのバージョン管理情報を保持するメタバージョンへのポインタを格納する属性を持たせ、該メタバージョンには、上記オブジェクトのバージョンを要素とする集合オブジェクトへのポインタを格納する属性ならびに該集合オブジェクトの要素を代表する代表バージョンへのポインタを格納する属性を持たせておき、オブジェクトの生成要求があった場合には、該要求により生成したオブジェクトの上記メタバージョンへのポインタを格納する属性に、該メタバージョンが存在しない旨を設定しておき、該オブジェクトのバージョン化要求があった場合には、該オブジェクトに対する上記メタバージョンを生成し、該オブジェクトを複写して該オブジェクトの最初のバージョンとなるオブジェクトを生成し、該オブジェクトにはジェネリックインスタンスであることを示すタグを付加し、該オブジェクトならびに該バージョンのメタバージョンへのポインタを格納する属性には、該オブジェクトに対して生成しておいた上記メタバージョンへのポインタを示す値を設定し、当該メタバージョンについては、該バージョンを要素とする集合オブジェクトを生成して、当該集合オブジェクトへのポインタを示す値と該バージョンへのポインタを示す値を、それぞれ、バージョンを要素とする集合オブジェクトへのポインタを格納する属性、集合オブジェクトの要素を代表する代表バージョンへのポインタを格納する属性に設定しておき、該オブジェクトに新たにバージョンを追加する場合は、該オブジェクトのメタバージョンに設定されている集合オブジェクトに、当該バージョンを要素として追加し、メッセージを処理する場合には、そのメッセージのレシーバに指定されているオブジェクトについて、該オブジェクトがジェネリックインスタンスであることを表すタグがあるか否かを判定し、該オブジェクトがジェネリックインスタンスでない場合は、該オブジェクトをレシーバとしてメッセージを処理し、該オブジェクトがジェネリックインスタンスである場合は、該オブジェクトのメタバージョンを得、該メタバージョンから代表バージョンを得、当該代表バージョンをレシーバとして

9

メッセージを処理する。

【0021】

【作用】本発明によるオブジェクト指向データベースシステム及びバージョン管理方法によれば、当該オブジェクト指向データベースシステムを用いて構築されるアプリケーションプログラムは、該プログラムが扱うオブジェクトのバージョン管理のために必要となるデータ構造（例えば、導出履歴管理のための木構造など）とそれを扱う操作をクラスとして定義し、該オブジェクト指向データベースシステムが提供するメタバージョンクラス、バージョンマネジャクラスの定義を継承し、さらに上記により定義した固有のデータを操作する機能を追加することにより、それぞれのクラスを該アプリケーションプログラム用にカスタマイズし、オブジェクトのバージョン管理を行う前に、カスタマイズしたバージョンマネジャクラスに、インスタンスを生成しておく必要がある。

【0022】該アプリケーションプログラムは、オブジェクトのバージョン管理が必要となった場合、該オブジェクトのためのメタバージョンをメタバージョンクラスのインスタンスとして生成し、該オブジェクトと該メタバージョンをパラメタとして、予め生成しておいたバージョンマネジャクラスのインスタンスにバージョン化を要求する。この結果、該オブジェクトのジェネリックインスタンスと最初のバージョンが生成される。このバージョン化の機能は、該アプリケーションプログラム用にカスタマイズされたバージョンマネジャクラスならびにメタバージョンクラスが、本発明が提供するバージョンマネジャクラスならびにメタバージョンクラスから継承する機能である。もちろん、バージョン化の際必要となる該アプリケーション固有の処理が追加されていても良い。

【0023】上記オブジェクト指向データベースシステムは、オブジェクトに送られてきたメッセージを処理する場合、該オブジェクトがジェネリックインスタンスであるか否かを、オブジェクトのバージョン化の際付加するタグの有無によって判定し、該オブジェクトがジェネリックインスタンスでない場合は、該オブジェクトをそのままメッセージのレシーバとして該メッセージが処理され、該オブジェクトがジェネリックインスタンスである場合は、該オブジェクトの参照するメタバージョンを得、さらにこのメタバージョンから該オブジェクトの代表バージョンを得、当該代表バージョンをレシーバとして上記メッセージを処理する。この結果、バージョンを持ったオブジェクトに対する明示的参照ならびに暗黙的参照が実現される。

【0024】上記によると、バージョン化要求のないオブジェクトは、対応するメタバージョンを持たないこと、バージョン管理のためにユーザ定義のクラスに余分に付加されるインスタンス変数やメソッドは、コンパイル時にバージョン化オブジェクトクラスから継承する1

10

つの変数と2つのメソッドだけであることから、メモリの浪費ならびにメッセージ処理のためのメソッドの探索に関するオーバヘッドが少ない。また、本発明によるオブジェクト指向データベースシステムを用いて構築されるアプリケーションプログラムは、本発明が提供するバージョンマネジャクラスならびにメタバージョンクラスの定義に、必要なデータ構造や操作を付加することにより、該アプリケーション固有のバージョン管理機能を構築することができる。

【0025】

【実施例】以下、本発明の実施例を図面を用いて説明する。図1は、本実施例の全体構成を示す図である。計算機システム10は、その上でオブジェクトサーバ20や、該オブジェクトサーバ20のクライアント30が動作する。オブジェクトサーバ20は、通信処理部21、メッセージ処理部22、メモリ管理部23、ストレージ管理部24から構成される。メッセージ処理部22は、オブジェクトに定義されている手続きを実行する。通信処理部21は、オブジェクトサーバ20とクライアント30間の処理要求やデータの遣り取りを制御する。ストレージ管理部24は、オブジェクトサーバ20が処理するオブジェクトを2次記憶装置内のオブジェクトデータベース40上で管理する。メモリ管理部23は、メモリ上のオブジェクト領域60内のオブジェクトを管理する。メッセージ処理に必要なオブジェクトがオブジェクト領域60上にない場合など、必要に応じてオブジェクトデータベース40上のオブジェクトはオブジェクト領域60にロードされる。オブジェクトデータベース40中のシステムライブラリ50には、オブジェクトサーバ20が予め準備したオブジェクトが存在する。また、、オブジェクトデータベース40中のアプリケーションプログラムライブラリ70には、該アプリケーションプログラムを構成するオブジェクトならびに該アプリケーションプログラムの実行により生成されたオブジェクトが存在する。バージョン化オブジェクトクラス51、メタバージョンクラス52、バージョンマネジャクラス53は、本発明を実現するためにオブジェクトサーバ20が予め準備するクラスオブジェクトである。

【0026】図2は、本実施例の動作の概要を示す図である。オブジェクトサーバ20は、自分自身を初期化した後、クライアント30からのコネクト要求を待つ。コネクト要求があれば、該クライアントに対する子プロセスを生成する。親プロセスは、さらに他のクライアントからのコネクト要求を待つ。生成された子プロセスでは、クライアント30からの処理要求を待ち、クライアント30からの処理要求を受け取ると、これを処理し、その結果をクライアントへ返し、次の処理要求を待つという動作を繰り返す。クライアント30から終了要求を受け取ると、子プロセスを削除しクライアント30との通信を終了する。

【００２７】本実施例のオブジェクトサーバ２０を用いた場合のアプリケーションプログラムのプログラミングは、まず問題領域をクラスという抽象概念の集まりに分けてクラスをつくるところから始まる。そして、クラスが揃えば、クラスにインスタンスオブジェクトを生成することができる。このクラスオブジェクトやインスタンスオブジェクトが、図１のアプリケーションプログラムライブラリ７０を構成する。クライアント３０からの要求をトリガに該アプリケーションプログラムが起動されると、以後は上記クラスオブジェクトやインスタンスオブジェクトにメッセージを送ることによってプログラムが進行する。

【００２８】クラスでは、そのクラスのインスタンスの内部状態を表す値を格納するインスタンス変数と、インスタンスに対して適用可能な手続きであるメソッドを定義する必要がある。このとき、より上位のクラスであるスーパクラスを指定することにより、そのスーパクラスに定義されているインスタンス変数やメソッドを継承することができる。例えば、人クラスの場合、人Ａ、人Ｂ、人Ｃがインスタンスであり、人の性別、生年月日等がインスタンス変数であり、男、昭和１年１月１日等がインスタンス変数の内部状態であり、生年月日から年齢を得るための手続きがメソッドであり、メソッドはプログラミングされている。メソッドは、メッセージによって起動される。メッセージには、そのメッセージのレシーバとなるオブジェクト、起動するメソッドを示すメソッド名、及びメソッドに対するパラメタ値を含んでいる。

【００２９】クラスの記述内容を示す例を図３に示す。この例は、「研究メモ」というクラスの記述例である。研究メモクラス３０１は、スーパクラス定義３０２により、ドキュメントクラス４０１をスーパクラスに指定し、インスタンス変数定義３０３により、メモ番号３０４、関連メモ３０５、目的３０６、本文３０７、挿入図３０８をインスタンス変数に持ち、メソッド定義３０９により、グループ内回覧３１０をメソッドに持つ。

【００３０】同様に、研究メモクラス３０１のスーパクラスであるドキュメントクラス４０１の記述内容を示す例を図４に示す。インスタンス変数定義４０３により、タイトル４０４、ドキュメント種類４０５、キーワード４０６、作成者４０７、審査者４０８、承認者４０９をインスタンス変数に持ち、メソッド定義４１０により、回覧４１１、審査４１２、承認４１３をメソッドに持つ。

【００３１】スーパクラス定義４０２により、ドキュメントクラス４０１のスーパクラスに指定されているオブジェクトクラスは、全てのオブジェクトが持つインスタンス変数やメソッドの定義を持つクラスである。例えば、インスタンスを生成するメソッド等をこのクラスに持たせ、明示的にスーパクラスに指定しなくても、全てのクラスにこの定義を継承させることにより、全てのオブジェクトが、システムが想定したオブジェクトとしての性質を持つようにすることができる。本発明は上記クラスのインスタンス変数やメソッドの詳細には無関係なので、ここではこれ以上言及しない。

【００３２】通常、オブジェクト指向データベースシステムは、それぞれのクラス定義言語とそのコンパイラを持っている。コンパイラは、図３や図４に示したような内容を持つクラス定義文をパラメタとして入力し、予め定められた形式のクラスオブジェクトを生成する。インスタンス変数やメソッドの定義ならびにスーパクラスからの継承もコンパイラによって処理され、クラスオブジェクトの中に埋め込まれる。本発明の好ましい構成では、オブジェクトサーバ２０は、コンパイラをクラスに適用可能なメソッドとして、システムライブラリ５０に用意する。

【００３３】以下、図５と図６を用いて、コンパイラが生成する一般的なクラスオブジェクトの内容や継承の処理の仕方について説明する。図３の研究メモクラスをコンパイルした結果生成されるクラスオブジェクトの概要を図５に示す。クラスオブジェクト中のインスタンス変数情報には、スーパクラスであるドキュメントクラス４０１に定義されているインスタンス変数４０４～４０９のタイトル、ドキュメント種類等が継承の結果として取り込まれている。同様に、インスタンスメソッド情報にもスーパクラスであるドキュメントクラス４０１に定義されているメソッド４１１～４１３の回覧、審査、承認が継承の結果として取り込まれている。ただし、継承したメソッドの実行コードは、ドキュメントクラス４０１のメソッドの実行コードを共有している。

【００３４】インスタンスメソッド情報については、図６に示すように、スーパクラスのメソッドを取り込まない方法も考えられる。図５は、スーパクラスからの継承がコンパイル時に全て処理されている状態であり、図６は、インスタンス変数については継承がコンパイル時に処理されているが、メソッドについては処理されていない状態である。図６のような場合は、実際にメソッドを探索する時にスーパクラス（この場合、ドキュメント）へも探索に行くことによって継承が実現できる。本発明はクラスオブジェクトや継承の実現方法には無関係なので、ここではこれ以上言及しない。

【００３５】図７～図９は、オブジェクトサーバ２０が予めシステムライブラリ５０の中に用意しているバージョン化オブジェクトクラス（図７中では、ＶｅｒｓｉｏｎｅｄＯｂｊｅｃｔ）５１、メタバージョンクラス（図８中では、ＭｅｔａＶｅｒｓｉｏｎ）５２、バージョンマネジャクラス（図９中では、ＶｅｒｓｉｏｎＭａｎａｇｅｒ）５３について、それぞれのクラスオブジェクトの内容を示したものである。

【００３６】図７で、@ｍｅｔａＶｅｒｓｉｏｎ７１

は、オブジェクトのバージョン管理情報を保持するメタバージョンへのポインタを格納するためのインスタンス変数、ｇｅｔＭｅｔａＶｅｒｓｉｏｎ７２は、@ｍｅｔａＶｅｒｓｉｏｎ７１の値を返すメソッド、ｓｅｔＭｅｔａＶｅｒｓｉｏｎ：７３は、パラメタとして受け取ったメタバージョンを@ｍｅｔａＶｅｒｓｉｏｎ７１に設定するメソッドである。

【００３７】図８で、@ｖｅｒｓｉｏｎＳｅｔ８１は、複数のバージョンを要素とする集合オブジェクトへのポインタを格納するためのインスタンス変数、@ｇｅｎｅｒｉｃ（該インスタンス変数に付けた名称である）８２は、ジェネリックインスタンスへのポインタを格納するためのインスタンス変数、@ｃｕｒｒｅｎｔ（該インスタンス変数に付けた名称である）８３は、集合オブジェクトの要素である複数のバージョンを代表する代表バージョンへのポインタを格納するためのインスタンス変数、ｇｅｔＶｅｒｓｉｏｎＳｅｔ８４は、@ｖｅｒｓｉｏｎＳｅｔ８１の値を返すメソッド、ｓｅｔＶｅｒｓｉｏｎＳｅｔ：８５は、パラメタとして受け取った集合オブジェクトを@ｖｅｒｓｉｏｎＳｅｔ８１に設定するメソッド、ｇｅｔＧｅｎｅｒｉｃ８８は、@ｇｅｎｅｒｉｃ８２の値を返すメソッド、ｓｅｔＧｅｎｅｒｉｃ：８９は、パラメタとして受け取ったジェネリックインスタンスを@ｇｅｎｅｒｉｃ８２に設定するメソッド、ｇｅｔＣｕｒｒｅｎｔ８６は、@ｃｕｒｒｅｎｔ８３の値を返すメソッド（ｇｅｔＣｕｒｒｅｎｔをセレクトするメッセージを送ると結果として代表バージョンを得ることになる）、ｓｅｔＣｕｒｒｅｎｔ：８７は、パラメタとして受け取ったバージョンを@ｃｕｒｒｅｎｔに設定するメソッドである。

【００３８】図９で、ｖｅｒｓｉｏｎｉｎｇ：ｗｉｔｈ：９１は、第１パラメタで受け取ったバージョン化対象オブジェクトと第２パラメタで受け取ったメタバージョンに基いて、オブジェクトをバージョン化するメソッドである。

【００３９】本発明のオブジェクトサーバ２０が用意しているコンパイラ５４は、クライアント３０がクラスのコンパイルを要求した場合、上記バージョン化オブジェクトクラス５１の定義を、コンパイル要求されたクラスに継承させる。したがって、図３の研究メモクラスのコンパイル要求があった場合、コンパイラ５４は、図１０に示すようなクラスオブジェクトを生成する。バージョンマネジャクラス５３にインスタンスを生成する（以下、このインスタンスを単にバージョンマネジャと呼ぶ。）と、このバージョンマネジャをレシーバとして図９に示したインスタンスメソッドを実行することができる。

【００４０】図１１を用いて、オブジェクトのバージョン化を行うバージョンマネジャクラス５３のｖｅｒｓｉｏｎｉｎｇ：ｗｉｔｈ：メソッド９１の処理について詳

細に説明する。まず、第１パラメタで与えられるオブジェクト、すなわちバージョン化対象オブジェクト（ＯＢＪ）にｇｅｔＭｅｔａＶｅｒｓｉｏｎをセレクタとするメッセージを送信し（ステップ１１１）、リターン値によりメタバージョンが既に存在しているかどうかを判定する（ステップ１１２）。メタバージョンが既に存在している場合は、バージョン化済みである旨を結果として返し（ステップ１１３）、処理を終了する。メタバージョンがまだ存在していない場合は、ＯＢＪを複写して新しいオブジェクト（ＶＥＲ）を生成し（ステップ１１４）、ＯＢＪにはジェネリックインスタンスであることを示すタグを付ける（ステップ１１５）。さらに、ＯＢＪならびにＶＥＲをレシーバとし、ｓｅｔＭｅｔａＶｅｒｓｉｏｎ：をセレクタとし、第２パラメタで与えられたメタバージョンをパラメタに指定したメッセージを送信して、このメタバージョンとの参照関係を設定し（ステップ１１６～ステップ１１７）、ステップ１１４で生成したＶＥＲをメッセージの送り手（関数のコール元に相当するに）返して（スッテプ１１８）、処理を終了する。

【００４１】バージョン化しないオブジェクトについては、メタバージョンやそれに係わる余分なオブジェクトは生成されないこと、また、バージョン管理に係わり全てのオブジェクトが一様に持たなければならないインスタンス変数は、バージョン化オブジェクトクラス５１から継承する@ｍｅｔａＶｅｒｓｉｏｎ７１だけであるということ等より、従来技術で述べた方法に比べてメモリの浪費が少ない。

【００４２】メッセージ処理部２２の処理について、図１２を用いて詳細に説明する。メッセージ処理部２２は、まず受け取ったメッセージ（ＭＳＧ）を解析して、レシーバに指定されているオブジェクト（ＯＢＪ）、実行すべきメソッドを指定しているセレクタ（ＳＥＬ）、メソッドに対するパラメタ（ＰＡＲＭ）を取り出す（ステップ１２１）。次に、ＯＢＪにジェネリックインスタンスであることを示すタグがあるかどうかを判定する（ステップ１２２）。ＯＢＪにタグがついている場合、ＯＢＪをレシーバとしてｇｅｔＭｅｔａＶｅｒｓｉｏｎをセレクタとするメッセージを送信し、ＯＢＪのメタバージョンを得る（ステップ１２４）。さらに、このメタバージョンをレシーバとしてｇｅｔＣｕｒｒｅｎｔをセレクタとするメッセージを送信し、代表バージョンを得（ステップ１２５）、現在の処理対象であるメッセージ（ＭＳＧ）のレシーバをＯＢＪからこの代表バージョンに置き換える（ステップ１２６）。次に、ＳＥＬに対応するメソッドを探索し（ステップ１２７）、得られたメソッドを実行する（ステップ１２８）。ステップ１２２においてＯＢＪにタグが無い場合、ＯＢＪをメッセージ（ＭＳＧ）のレシーバとし（ステップ１２３）、ステップ１２７～ステップ１２８を実行する。

15

【0043】上記により、バージョンに対する明示的参照ならびに暗黙的参照が実現されている。明示的参照は、オブジェクトXの任意のバージョンをX（i）（i＝1・・・n）とすると、X（i）を直接参照する（X（i）をレシーバとしてメッセージを送ること、X（i）が直接メッセージをうける）。暗黙的参照は、特定のバージョンではなくX（個々のバージョンに対してこちらをジェネリックインスタンスと呼んでいる）を参照する（メッセージはXに送るが、内部的にはその時点の代表バージョンがメッセージを受ける）。図12では、ステップ122で明示的参照か暗黙的参照かを判定し（ジェネリックインスタンスにタグを付けておき、ジェネリックインスタンスをレシーバとするものは暗黙的参照としている）、明示的参照の場合はステップ123を、暗黙的参照の場合はステップ124～126を実行している。バージョン管理のための操作を、ユーザが定義した個々のクラスに継承させていないため、図12のステップ127で探索対象となるメソッドのうち、バージョン管理に係わるメソッドは、バージョン化オブジェクトクラス51から継承する2つのメソッドのみである。また、メタバージョンを介するのは暗黙的参照解決のために代表バージョンを必要とする場合のみである。したがって、それぞれのクラスにユーザが定義したメソッドの実行に対して、バージョン管理機能実現に伴うオーバヘッドは、従来技術で述べた方法に比べて小さい。

【0044】以下では、本発明によるオブジェクトサーバ20に基いて、クライアント30のより実際的なバージョン管理機能がどのように実現できるかについて述べる。例として、次のような仕様を持つバージョン管理機能の実現を考える。

（1）最初のバージョンを除き、新たなバージョンは既存のバージョンから生成する。

（2）上記バージョンの導出履歴を管理する。

【0045】図13～図15は、アプリケーションプログラムの開発者が、上記バージョン管理機能の実現のために用意するクラスオブジェクトを示す図である。図13のVersionTreeクラス130は、バージョンの導出履歴を管理するための属性や操作を記述したクラスである。VersionTreeクラス130には、図16に示すようなトリー構造のデータを持つオブジェクトへのポインタを格納するためのインスタンス変数@rootNode131、与えられたノードに子ノードを追加するaddNode：to：メソッド132、与えられたノードの親ノードを返すgetPrecedeNode：メソッド133、与えられたノードの子ノードのリストを返すgetSucceedNodeList：メソッド134を用意する。図16中の各ノードは1つ1つのバージョンに対応し、図中の矢印は、始点のバージョンから終点のバージョンが導出されているという関係を表す。VersionTreeクラス130は、通常のプログラミングの場合と同様、オブジェクト指向データベースシステムが予め用意している、整数やリストやアレイ等の基本クラスによって実現できるレベルのクラスであるため、これ以上言及しない。

16

【0046】図14のCustomizedMetaVersionクラス140は、メタバージョンクラス52をスーパクラスとし、メタバージョンクラス52からインスタンス変数81～83、メソッド84～89を継承する。CustomizedMetaVersionクラス140固有のインスタンス変数として、導出履歴を保持するVersionTreeクラス130のインスタンスへのポインタを格納するためのインスタンス変数@versionTree141と、この変数の値を返すgetVersionTreeメソッド142、この変数に値を設定するsetVersionTree：メソッド143、バージョンを追加するためのaddDerivedVersion：from：メソッド144を持つ。

【0047】addDerivedVersion：from：メソッド144では、まず、getVersionSet84を用いてインスタンス変数@versionSet81の値から集合オブジェクトを得、この集合オブジェクトに第1パラメタで与えられたバージョンを追加する。次に、getVersionTree142を用いてインスタンス変数@versionTree141の値からバージョントリーインスタンスを得、このバージョントリーインスタンスに、addNode：to：メソッド132を用いて、第1パラメタで与えられたバージョンが第2パラメタで与えられたバージョンの子バージョンとなるように新しいノードを追加する。

【0048】図15のCustomizedVersionManagerクラス150は、バージョンマネジャクラス53をスーパクラスとし、バージョンマネジャクラス53からメソッド91を継承する。CustomizedVersionManagerクラス150固有のメソッドとして、与えられたオブジェクトをバージョン化するversioning：メソッド151、与えられたバージョンから子バージョンを導出するderiveFrom：メソッド152、与えられたバージョンの親バージョンを返すメソッドgetParent：153、与えられたバージョンの子バージョンのリストを返すgetChildren：メソッド154を持つ。

【0049】CustomizedVersionManagerクラス150の、オブジェクトのバージョン化処理を行うversioning：メソッド151の処理について、図17を用いて説明する。versioning：メソッド151では、まず、CustomizedMetaVersionクラス140にインスタンス（META）を生成する（ステップ171）。次

に、ｖｅｒｓｉｏｎｉｎｇ：メソッド１５１のレシーバ自身（ＣｕｓｔｏｍｉｚｅｄＶｅｒｓｉｏｎＭａｎａｇｅｒクラスのインスタンス、すなわちバージョンマネジャ）をレシーバとし、パラメタで与えられるオブジェクト、すなわちバージョン化が要求されているオブジェクト（ＯＢＪ）を第１パラメタ、ステップ１７１にて生成したメタバージョン（ＭＥＴＡ）を第２パラメタとして、ｖｅｒｓｉｏｎｉｎｇ：ｗｉｔｈ：をセレクタとするメッセージを送信する（ステップ１７２）。これにより、図１１に示した処理が実行される。ステップ１７２の結果から、既にメタバージョンが存在しているかどうか、すなわちバージョン化済みであるかどうかを判定し（ステップ１７３）、ＯＢＪが既にバージョン化済みであった場合、ステップ１７１で生成したメタバージョン（ＭＥＴＡ）を消去し（ステップ１７４）、ＯＢＪが既にバージョン化済みである旨を返して（ステップ１７５）処理を終了する。ＯＢＪがバージョン化されていない場合、まずＶｅｒｓｉｏｎＴｒｅｅクラス１３０に、ステップ１７２の結果リターンされる最初のバージョンがルートノードとなるようにバージョントリーインスタンス（ＴＲＥＥ）を生成する（ステップ１７６）。さらに、ＭＥＴＡをレシーバとし、ＴＲＥＥをパラメタとし、ｓｅｔＶｅｒｓｉｏｎＴｒｅｅ：をセレクタとするメッセージを送信し、このＭＥＴＡとＴＲＥＥとの関係を設定し（ステップ１７７）、ステップ１７２の結果リターンされる最初のバージョンを返し（ステップ１７８）て、処理を終了する。

【００５０】以下、ＣｕｓｔｏｍｉｚｅｄＶｅｒｓｉｏｎＭａｎａｇｅｒクラス１５０の他のメソッドについて説明する。図１８は、ｄｅｒｉｖｅＦｒｏｍ：メソッド１５２の処理フローを示す。まず、第１パラメタで与えられるオブジェクト（ＯＢＪ）を複写して、新しいバージョン（ＶＥＲ）を生成する（ステップ１８１）。ＯＢＪをレシーバとし、ｇｅｔＭｅｔａＶｅｒｓｉｏｎをセレクタとするメッセージを送信して、参照するメタバージョン（ＭＥＴＡ）を得る（ステップ１８２）。次に、ＭＥＴＡをレシーバとし、生成した子バージョンＶＥＲを第１パラメタ、ＯＢＪを第２パラメタとして、ａｄｄＤｅｒｉｖｅｄＶｅｒｓｉｏｎ：ｆｒｏｍ：を送信して、ＭＥＴＡに生成したＶＥＲの情報を追加する（ステップ１８３）。さらに、ＭＥＴＡをレシーバとし、生成した子バージョンＶＥＲをパラメタとして、ｓｅｔＣｕｒｒｅｎｔ：を送信し、ＶＥＲを代表バージョンに設定（ステップ１８４）し、ＶＥＲを返し（ステップ１８５）て処理を終了する。

【００５１】図１９は、ｇｅｔＰａｒｅｎｔ：メソッド１５３の処理フローを示す図である。第１パラメタで与えられたオブジェクト（ＯＢＪ）をレシーバとし、ｇｅｔＭｅｔａＶｅｒｓｉｏｎをセレクタとするメッセージを送信して、参照するメタバージョン（ＭＥＴＡ）を得る（ステップ１９１）。さらにそのＭＥＴＡをレシーバとして、ｇｅｔＶｅｒｓｉｏｎＴｒｅｅをセレクタとするメッセージを送信し、参照するバージョントリーインスタンス（ＴＲＥＥ）を得（ステップ１９２）、このバージョントリーインスタンスＴＲＥＥから第１パラメタで与えられるバージョンの親バージョンを得てこれを返す（ステップ１９３）。

【００５２】図２０は、ｇｅｔＣｈｉｌｄｒｅｎ：メソッド１５４の処理フローを示す図である。第１パラメタで与えられたオブジェクト（ＯＢＪ）をレシーバとし、ｇｅｔＭｅｔａＶｅｒｓｉｏｎをセレクタとするメッセージを送信して、参照するメタバージョン（ＭＥＴＡ）を得る（ステップ２０１）。さらにそのＭＥＴＡをレシーバとし、ｇｅｔＶｅｒｓｉｏｎＴｒｅｅをセレクタとするメッセージを送信して、参照するバージョントリーインスタンス（ＴＲＥＥ）を得（ステップ２０２）、このバージョントリーインスタンスＴＲＥＥから、第１パラメタで与えられたバージョンの子バージョンのリストを得てこれを返す（ステップ２０３）。

【００５３】上記ＣｕｓｔｏｍｉｚｅｄＶｅｒｓｉｏｎＭａｎａｇｅｒクラス１５０にインスタンスを生成すると、このインスタンスは、クライアント３０に固有のバージョン管理機能を持ったバージョンマネジャとして動作する。すなわち、このインスタンスをレシーバとして、上記ＣｕｓｔｏｍｉｚｅｄＶｅｒｓｉｏｎＭａｎａｇｅｒクラス１５０に定義されているメソッド９１、１５１～１５４が実行可能である。上記の例で示したように、本発明によるオブジェクトサーバ２０を用いると、応用分野にマッチしたバージョン管理機能の実現が容易になるという効果もある。

【００５４】

【発明の効果】本発明によれば、バージョン化要求のないオブジェクトは、バージョン管理情報を保持するための余分なオブジェクトを持たないこと、バージョン管理のためにユーザ定義のクラスに余分に付加されるインスタンス変数は１つ、メソッドは２つであることから、バージョン管理のために生じるメモリの浪費や、メッセージ処理に対するオーバヘッドの少ないバージョン管理機能が実現できる。

【図面の簡単な説明】

【図１】実施例の全体構成を示す図である。

【図２】実施例の動作の概要を示す図である。

【図３】クラスの記述内容の例を示す図である。

【図４】クラスのスーパクラスの記述内容の例を示す図である。

【図５】クラスオブジェクトの内容を示す図である。

【図６】図５のクラスオブジェクトの内容に代わる内容を示す図である。

【図７】バージョン化オブジェクトクラスの内容を示す図である。

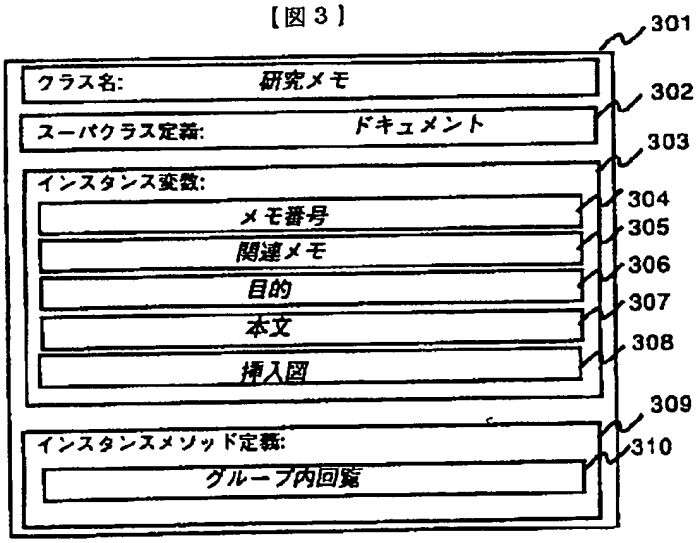
19　　20

【図8】メタバージョンクラスの内容を示す図である。
【図9】バージョンマネジャクラスの内容を示す図である。
【図10】本実施例のコンパイラが生成するクラスの内容を示す図である。
【図11】ｖｅｒｓｉｏｎｉｎｇ：ｗｉｔｈ：メソッドの処理フローを示す図である。
【図12】本実施例のメッセージ処理部の処理フローを示す図である。
【図13】ＶｅｒｓｉｏｎＴｒｅｅクラスの内容を示す図である。
【図14】ＣｕｓｔｏｍｉｚｅｄＭｅｔａＶｅｒｓｉｏｎクラスの内容を示す図である。
【図15】ＣｕｓｔｏｍｉｚｅｄＶｅｒｓｉｏｎＭａｎａｇｅｒクラスの内容を示す図である。
【図16】バージョン導出履歴のデータ構造を示す図である。
【図17】ｖｅｒｓｉｏｎｉｎｇ：メソッドの処理フローを示す図である。
【図18】ｄｅｒｉｖｅＦｒｏｍ：メソッドの処理フローを示す図である。
【図19】ｇｅｔＰａｒｅｎｔ：メソッドの処理フローを示す図である。
【図20】ｇｅｔＣｈｉｌｄｒｅｎ：メソッドの処理フローを示す図である。

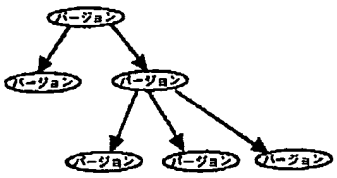
【符号の説明】

10　計算機システム
20　オブジェクトサーバ
21　通信処理部
22　メッセージ処理部
23　メモリ管理部
24　ストレージ管理部
30　クライアント
40　オブジェクトデータベース
50　システムライブラリ
51　バージョンオブジェクトクラス
52　メタバージョンクラス
53　バージョンマネジャクラス
54　コンパイラ
60　オブジェクト領域
70　アプリケーションプログラムライブラリ

【図3】

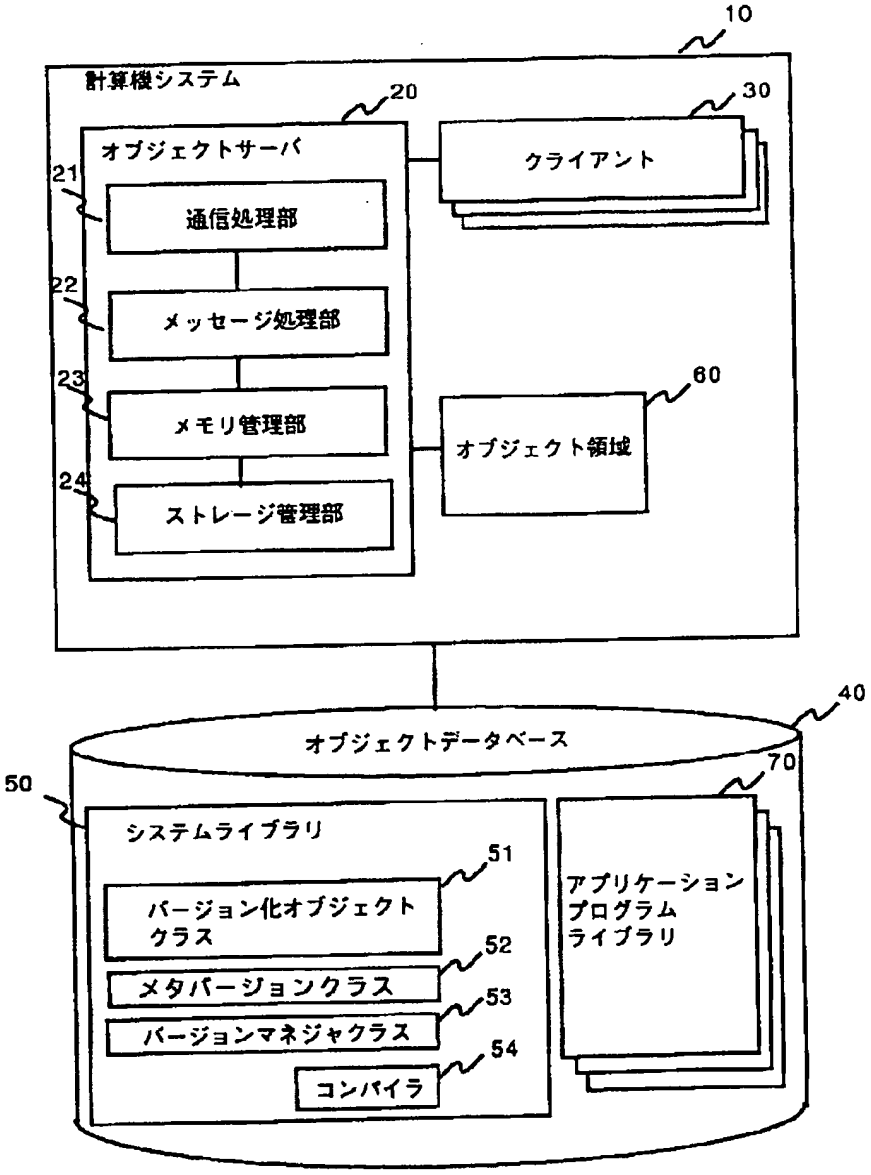


【図16】



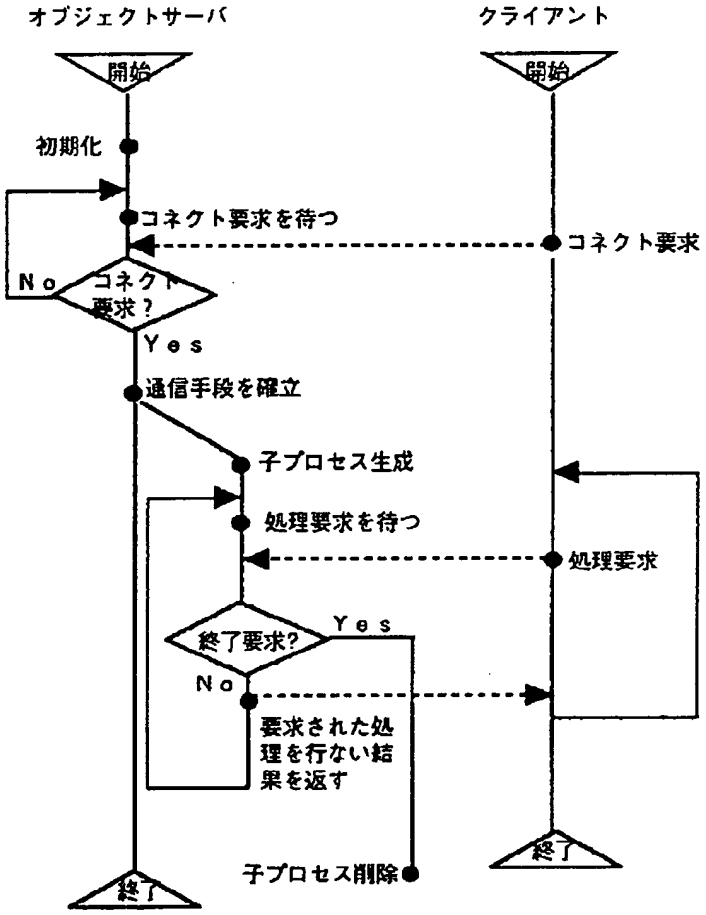
【図1】

［図１］


10
計算機システム
20
30
オブジェクトサーバ
クライアント
21
通信処理部
22
メッセージ処理部
23
メモリ管理部
60
オブジェクト領域
24
ストレージ管理部
40
オブジェクトデータベース
70
50
システムライブラリ
51
バージョン化オブジェクト
クラス
52
メタバージョンクラス
53
バージョンマネジャクラス
54
コンパイラ
アプリケーション
プログラム
ライブラリ

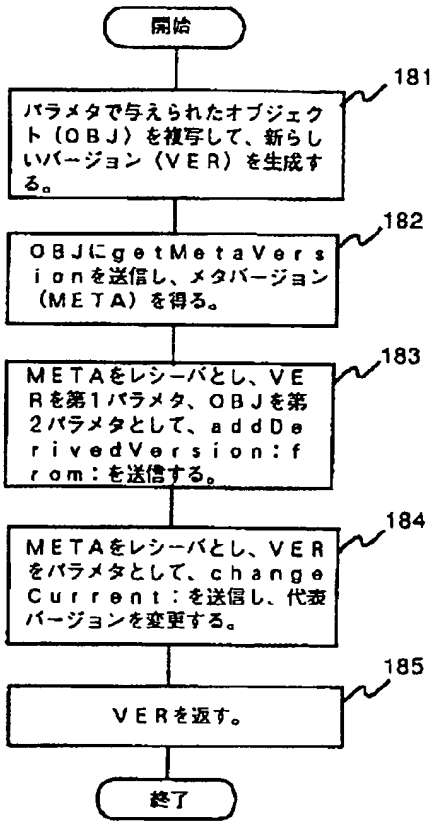
【図2】

【図2】



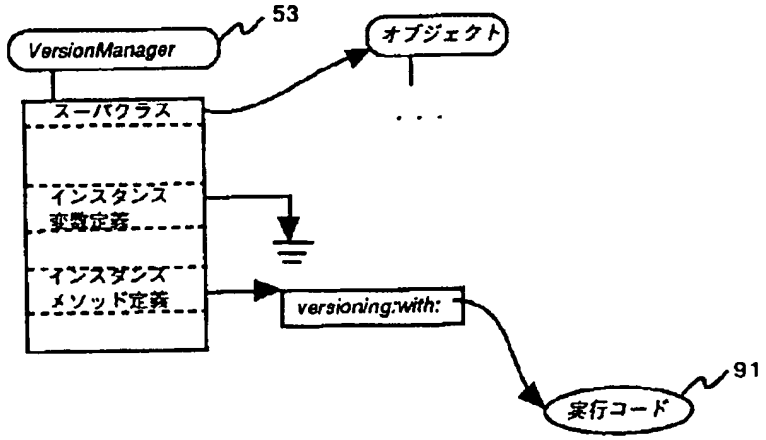
【図18】

【図18】



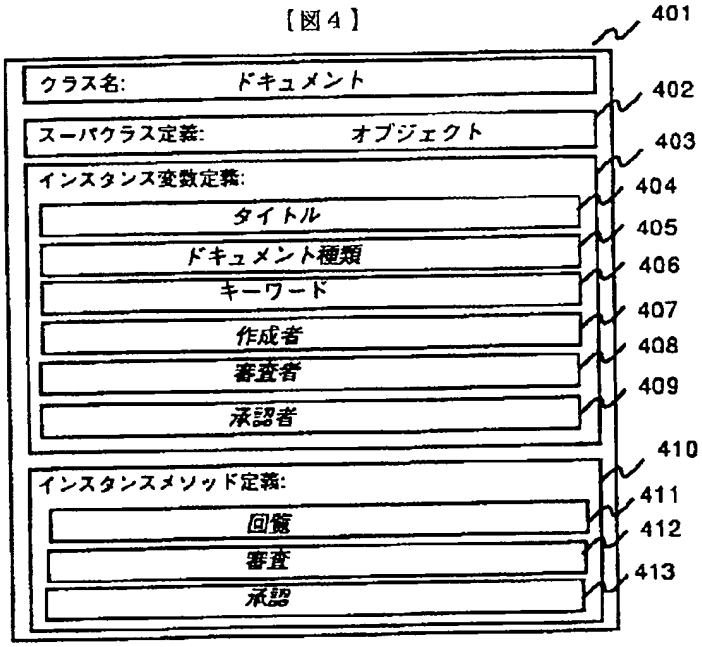
【図9】

【図9】



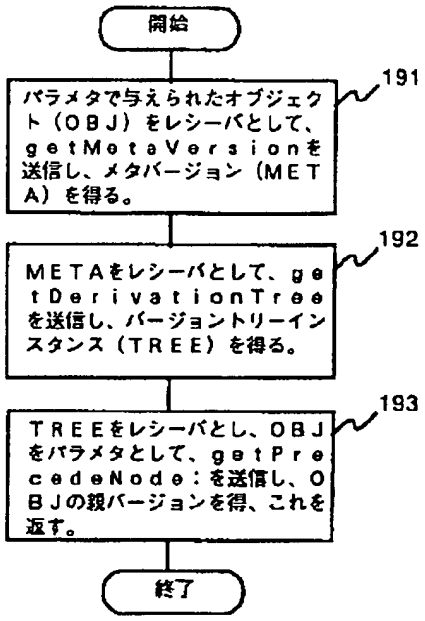
【図4】

【図4】



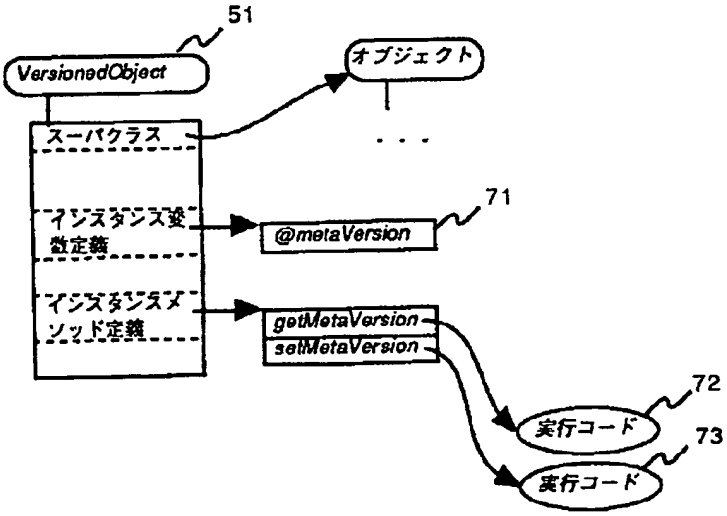
【図19】

【図19】



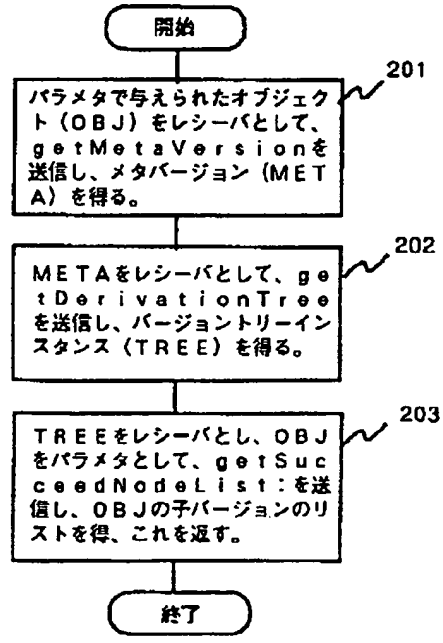
【図7】

【図7】



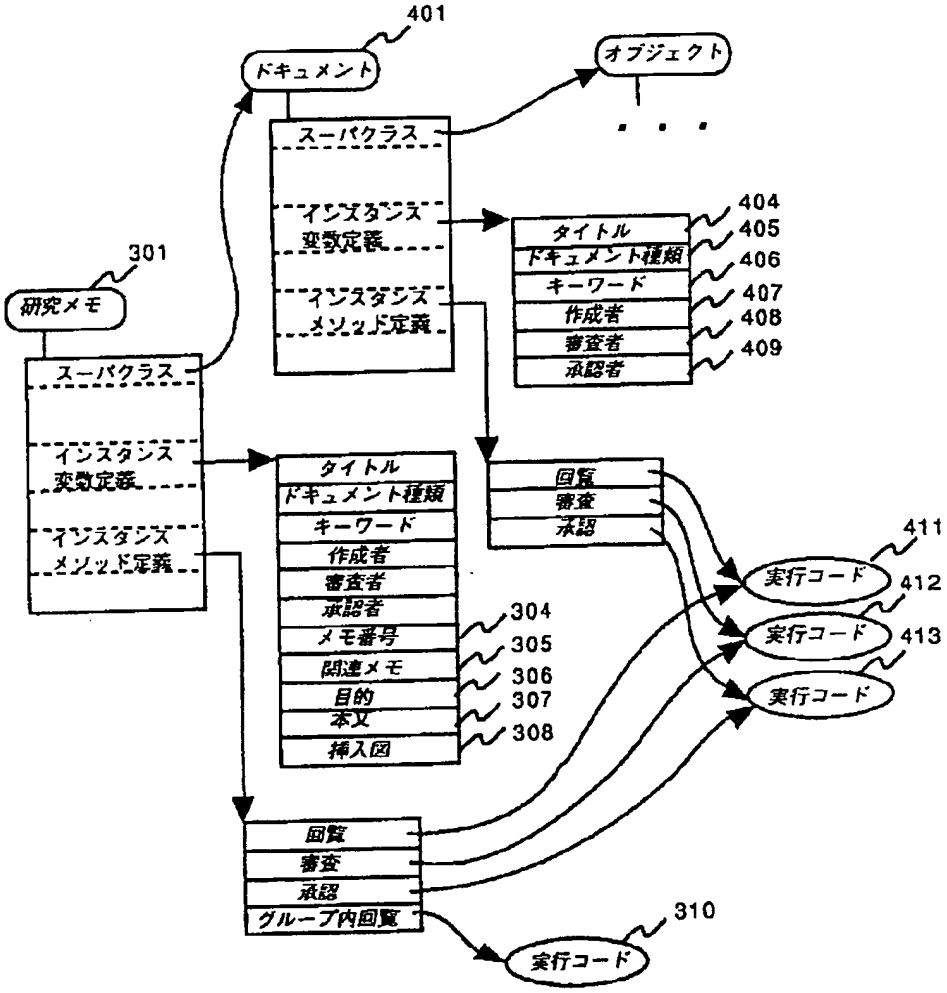
【図20】

【図20】



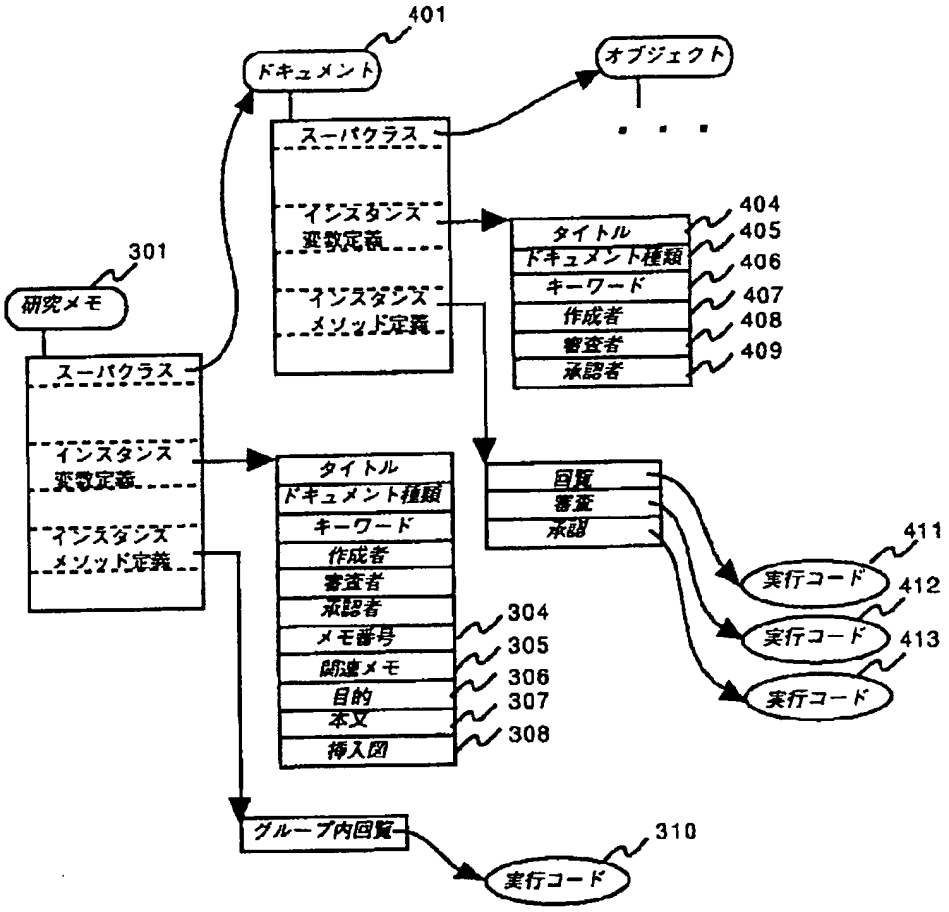
【図5】

［図５］


401
ドキュメント
オブジェクト
スーパクラス
インスタンス
変数定義
インスタンス
メソッド定義
404
タイトル
405
ドキュメント種類
406
キーワード
407
作成者
408
審査者
409
承認者
301
研究メモ
スーパクラス
インスタンス
変数定義
インスタンス
メソッド定義
タイトル
ドキュメント種類
キーワード
作成者
審査者
承認者
304
メモ番号
305
関連メモ
306
目的
307
本文
308
挿入図
回覧
審査
承認
411
実行コード
412
実行コード
413
実行コード
回覧
審査
承認
グループ内回覧
310
実行コード

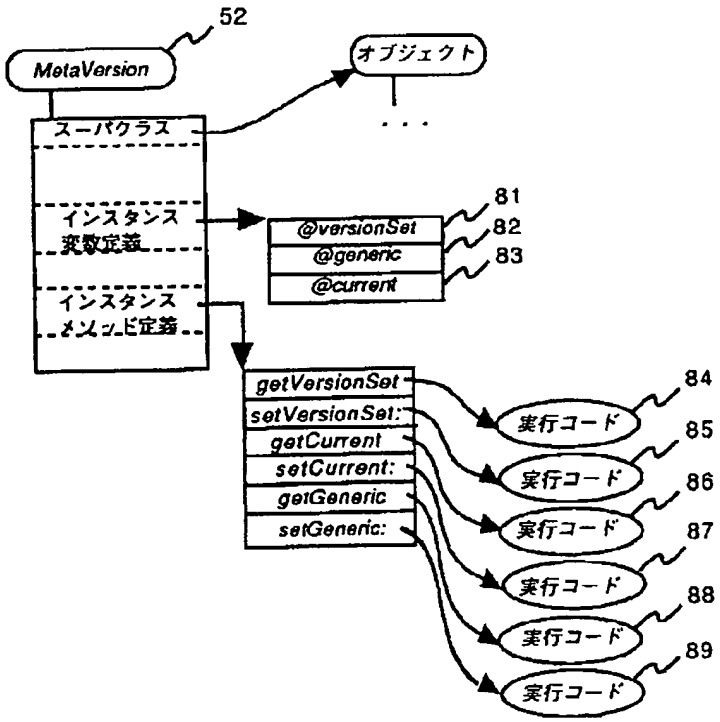
【図6】

【図6】


401
ドキュメント
オブジェクト
スーパクラス
インスタンス
変数定義
インスタンス
メソッド定義
404
405
406
407
408
409
タイトル
ドキュメント種類
キーワード
作成者
審査者
承認者
301
研究メモ
スーパクラス
インスタンス
変数定義
インスタンス
メソッド定義
タイトル
ドキュメント種類
キーワード
作成者
審査者
承認者
メモ番号
関連メモ
目的
本文
挿入図
304
305
306
307
308
回覧
審査
承認
411
412
413
実行コード
実行コード
実行コード
グループ内回覧
310
実行コード

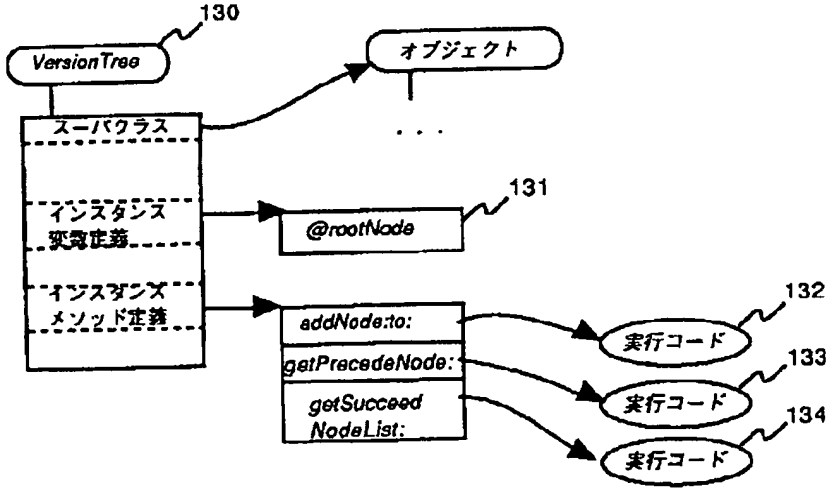
【図8】

［図８］


52
MetaVersion
オブジェクト
スーパクラス
インスタンス
変数定義
インスタンス
メソッド定義
81
@versionSet
82
@generic
83
@current
getVersionSet
setVersionSet:
getCurrent
setCurrent:
getGeneric
setGeneric:
84
実行コード
85
実行コード
86
実行コード
87
実行コード
88
実行コード
89
実行コード

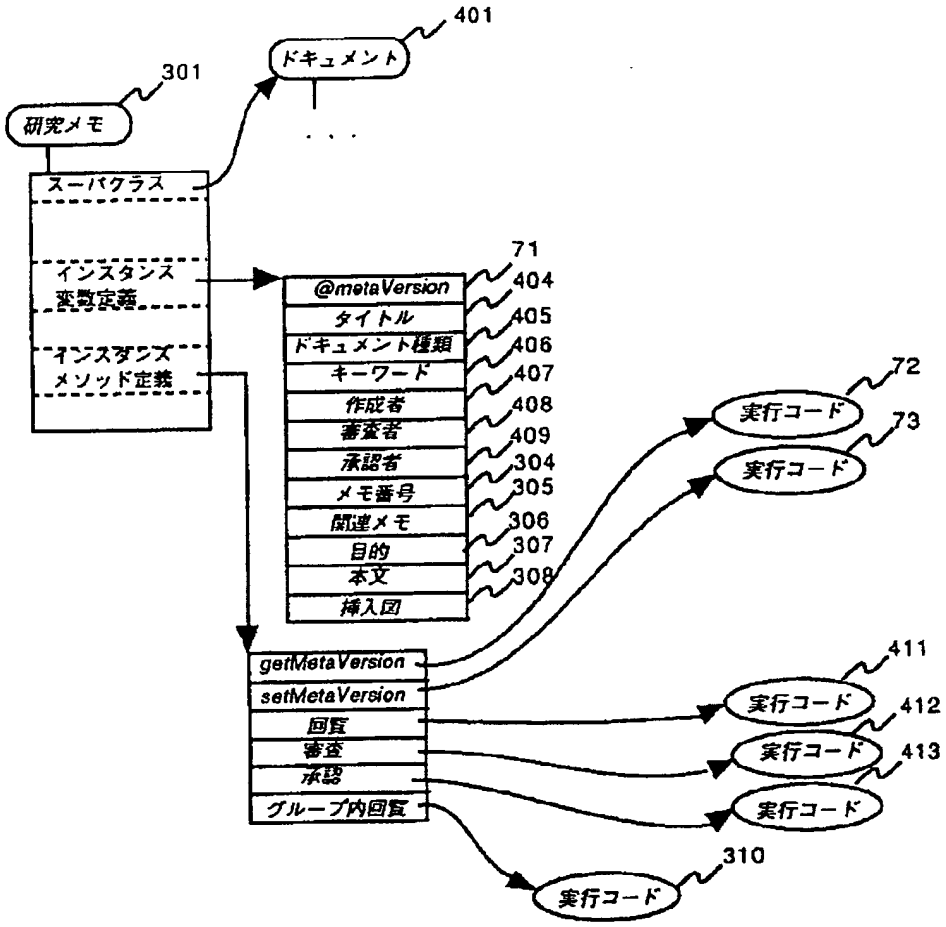
【図１３】

［図１３］


130
VersionTree
オブジェクト
スーパクラス
インスタンス
変数定義
インスタンス
メソッド定義
131
@rootNode
addNode:to:
getPrecedeNode:
getSucceed
NodeList:
132
実行コード
133
実行コード
134
実行コード

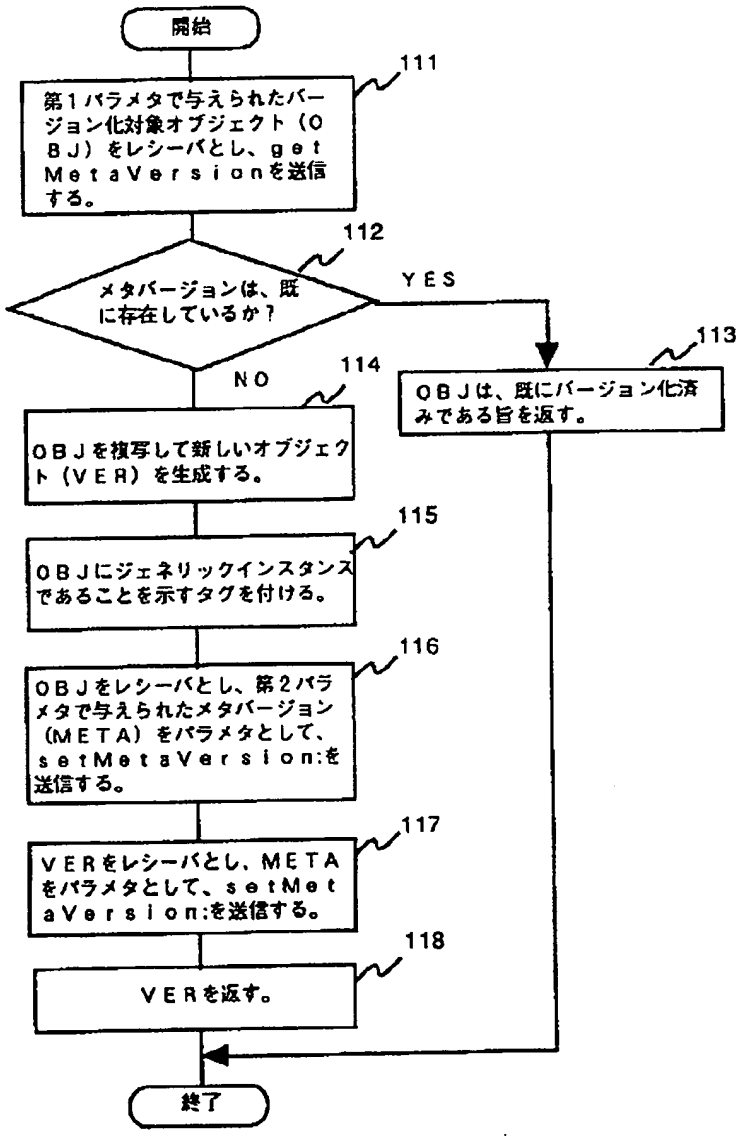
【図１０】

【図１０】


401
ドキュメント
301
研究メモ
スーパクラス
インスタンス変数定義
インスタンスメソッド定義
71
@metaVersion
404
タイトル
405
ドキュメント種類
406
キーワード
407
作成者
408
審査者
409
承認者
304
メモ番号
305
関連メモ
306
目的
307
本文
308
挿入図
72
実行コード
73
実行コード
getMetaVersion
setMetaVersion
回覧
審査
承認
グループ内回覧
411
実行コード
412
実行コード
413
実行コード
310
実行コード

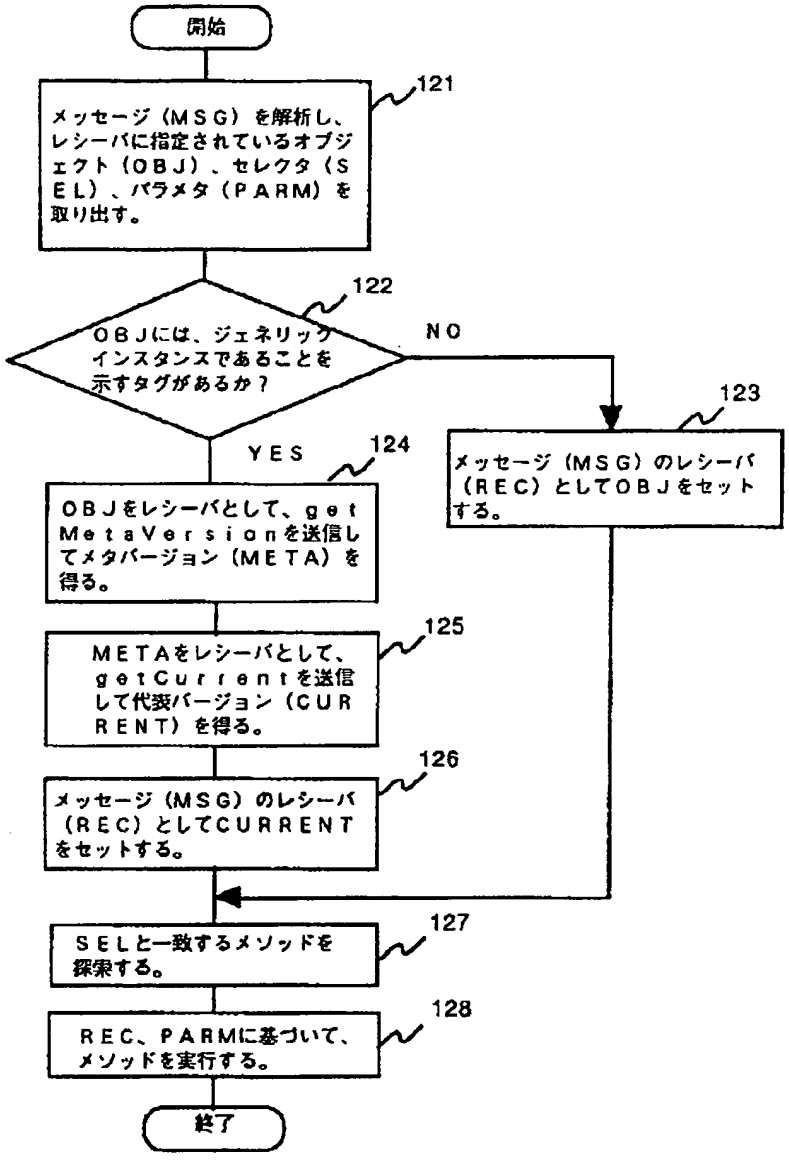
【図１１】

【図１１】


開始
111
第１パラメタで与えられたバージョン化対象オブジェクト（ＯＢＪ）をレシーバとし、ｇｅｔＭｅｔａＶｅｒｓｉｏｎを送信する。
112
メタバージョンは、既に存在しているか？
ＹＥＳ
113
ＯＢＪは、既にバージョン化済みである旨を返す。
ＮＯ
114
ＯＢＪを複写して新しいオブジェクト（ＶＥＲ）を生成する。
115
ＯＢＪにジェネリックインスタンスであることを示すタグを付ける。
116
ＯＢＪをレシーバとし、第２パラメタで与えられたメタバージョン（ＭＥＴＡ）をパラメタとして、ｓｅｔＭｅｔａＶｅｒｓｉｏｎ:を送信する。
117
ＶＥＲをレシーバとし、ＭＥＴＡをパラメタとして、ｓｅｔＭｅｔａＶｅｒｓｉｏｎ:を送信する。
118
ＶＥＲを返す。
終了

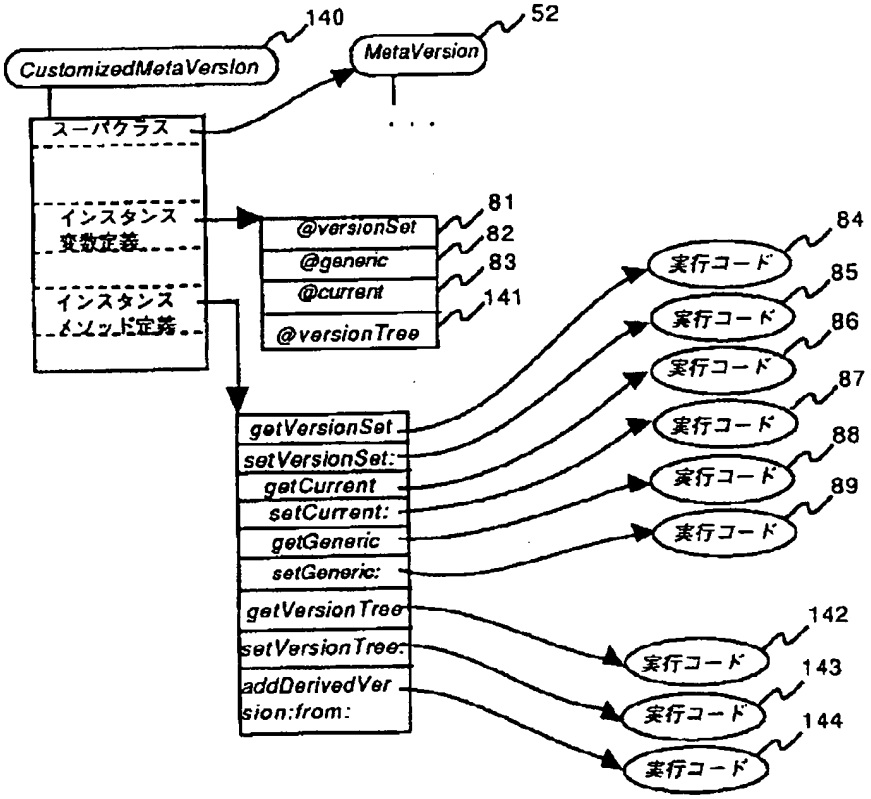
【図１２】

【図１２】


開始
メッセージ（ＭＳＧ）を解析し、レシーバに指定されているオブジェクト（ＯＢＪ）、セレクタ（ＳＥＬ）、パラメタ（ＰＡＲＭ）を取り出す。
121
ＯＢＪには、ジェネリックインスタンスであることを示すタグがあるか？
122
ＮＯ
ＹＥＳ
123
メッセージ（ＭＳＧ）のレシーバ（ＲＥＣ）としてＯＢＪをセットする。
124
ＯＢＪをレシーバとして、ｇｅｔＭｅｔａＶｅｒｓｉｏｎを送信してメタバージョン（ＭＥＴＡ）を得る。
125
ＭＥＴＡをレシーバとして、ｇｅｔＣｕｒｒｅｎｔを送信して代表バージョン（ＣＵＲＲＥＮＴ）を得る。
126
メッセージ（ＭＳＧ）のレシーバ（ＲＥＣ）としてＣＵＲＲＥＮＴをセットする。
127
ＳＥＬと一致するメソッドを探索する。
128
ＲＥＣ、ＰＡＲＭに基づいて、メソッドを実行する。
終了

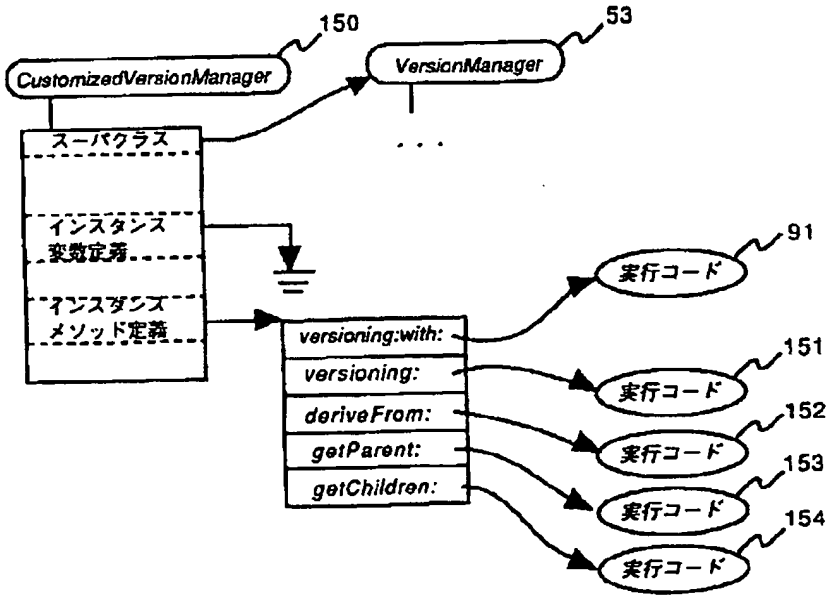
【図１４】

【図１４】



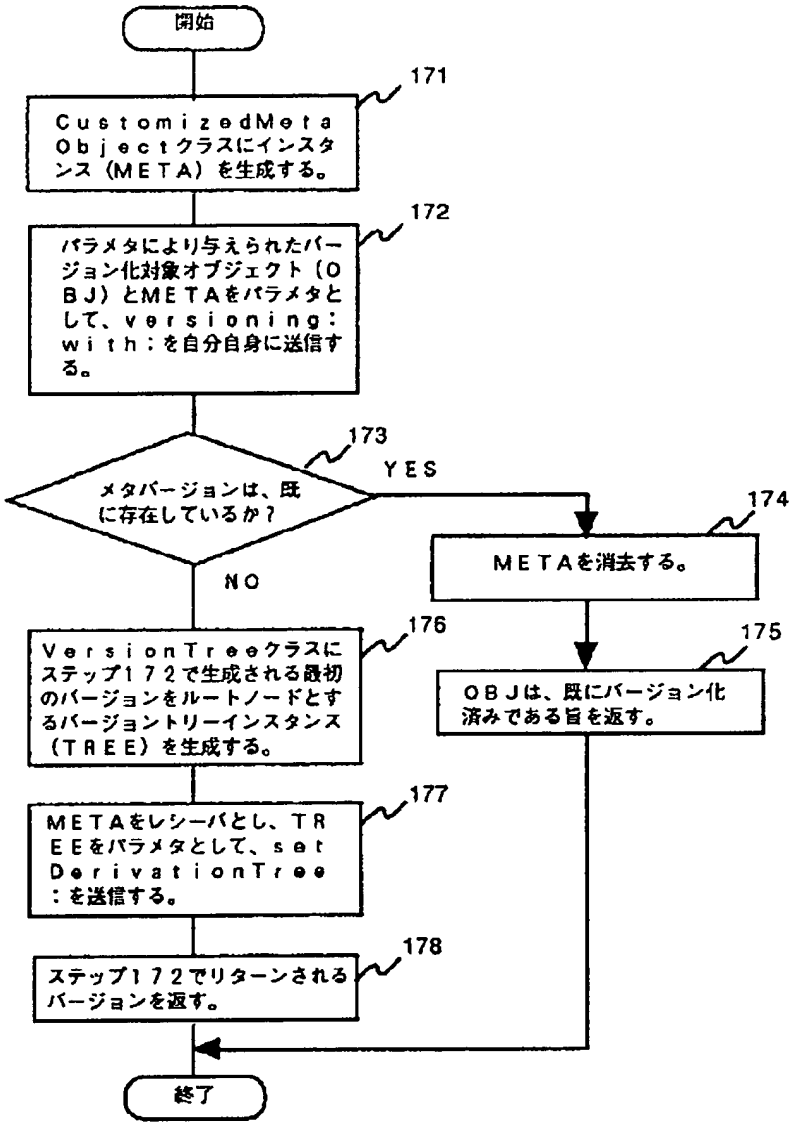
【図１５】

【図１５】



【図17】

【図17】



フロントページの続き

(72)発明者　古川　久美子
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(72)発明者　佐藤　和洋
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(72)発明者　丸山　剛男
神奈川県横浜市戸塚区戸塚町5030番地　株式会社日立製作所ソフトウェア開発本部内